Platforms for Serverless at the Edge: A Review

Nabil El Ioini[1(✉)], David Hästbacka[2], Claus Pahl[1], and Davide Taibi[2]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
{nabil.elioini,claus.pahl}@unibz.it
[2] Tampere University, Tampere, Finland
{david.hastbacka,davide.taibi}@tuni.fi

Abstract. The continuous demand for low latency, high reliability, and context-aware content has pushed the existing computational models to their limit. The cloud with its infinite resources can accommodate many of the existing scenarios, however, as new scenarios emerge in the IoT area, the cloud falls short. In this context, the Edge Computing model emerged as an extension to the cloud in support of low latency and high-performance applications, by placing part of cloud resources at the edge of the network, in close proximity to the data sources and applications. The goal of Edge Computing is to provide the same level of abstraction at the cloud but in a local context. However, since Edge Computing inherits many of the benefits provided by the cloud, it also inherits some of its drawbacks. One such limitation is the management overhead needed to set-up and continuously configure the Edge Computing applications. In the cloud space, this problem has been addressed using a new paradigm called serverless technology. Similarly, in the Edge Computing, few attempts are being developed to bring the concept of Serverless Computing at the edge. In this paper, we survey the main edge computing platforms that provide support for serverless computing comparing their characteristics and identifying issues and research directions.

Keywords: Serverless · Edge computing · FaaS · Function-as-a-Service · Technology review

1 Introduction

Edge computing, the new buzzword, has been gaining a lot of traction from developers and the industry. Companies are mainly interested in improving the performance of their systems and reducing the operational costs, by moving part of the cloud resources closer to the data sources. One major use case of edge computing is IoT applications [5,9]. In industrial settings, there are many scenarios such as condition monitoring and general production monitoring that would benefit from processing the often huge amounts of sensor data closer to the source [6]. Similarly in multi media applications, receiving data from nearby edge computing units [7] can have a huge impact on latency and the user experience.

C. Zirpins et al. (Eds.): ESOCC 2020 Workshops, CCIS 1360, pp. 29–40, 2021.
https://doi.org/10.1007/978-3-030-71906-7_3

Together with Edge computing, serverless computing (also known as Function-as-a-Service or FaaS) is now gaining more and more interest from companies. Cloud vendors such as AWS and Microsoft have hyped serverless almost everywhere, from practitioners' conferences to local events, to blog posts. They are promoting the idea of allowing companies to focus only on their business logic, while delegating all the operational tasks to the cloud provider. However, serverless is not just about the hype but has several benefits that enable companies to reduce costs and to focus more on the business logic of applications. Since the main goal of edge computing is to locate cloud resources closer to the end user, and serverless technology is one of these resources, it is legitimate to investigate how these two technologies can be combined and what are the benefits and limitations of such integration.

Different companies have started already combining the power of edge with the operational easiness of serverless providing edge platforms for deploying serverless functions. The adoption of serverless computing on edge nodes might help to reduce the computational time, and reduce network-related costs [2,3,8,11,13]. However, in this fast-growing market, it is still not clear how to benefit from the serverless capabilities on edge platforms, and especially how the available solutions on the market allow us to abstract from the hardware used at the edge using containers or serverless functions at the edge.

In order to help practitioners, and stimulate the discussion on this topic, we aim at comparing the most common edge platforms that provide serverless support, discussing pros, cons and highlighting open issues and research directions. Therefore, the main contributions of this work are:

- A list of the most common edge platforms that support serverless functions
- Comparison of the main characteristics of the platforms
- Identification of open issues and research directions

The results can be useful to the research community and to practitioners that can easily compare the different features of the platforms and understand how to better select edge platforms that support serverless.

The remainder of this paper is structured as follows. Section 2 introduces the background of Serverless and Edge Computing. Section 3 describes and compares the selected edge-computing platforms. Section 4 discusses the results and identify open issues and research directions. Finally, Sect. 5 draws conclusions and highlight future works.

2 Background

In this section, we introduce the two main technologies subject of this review, namely Serverless technology and Edge Computing.

2.1 Serverless

A few years ago, most companies were entirely responsible for the operations of their server-side applications, then the cloud enabled companies to outsource

part of the operations, renting virtual machines by the hour and paying as much concern to how much electricity our systems require as to how to use a mobile phone. However, the software systems remain as servers–discrete components that require allocation, provisioning, setting up, deploying, shutting down, ... In 2012 [10], developers started thinking about operating their systems instead of operating their servers, considering applications as workflows, distributed logic, and externally managed data stores. This way of working can be considered "serverless", not because no servers are running, but because developers do not need to think about them anymore.

In serverless, the cloud provider dynamically allocates and provisions servers. The code is executed in almost-stateless containers that are event-triggered, and ephemeral (may last for one invocation), and fully managed by the cloud provider [10].

However, the term serverless can be misleading. Serverless covers a wide range of technologies, that can be grouped into two categories: Backend-as-a-Service (BaaS) and Functions-as-a-Service (FaaS).

Backend-as-a-Service enables to replace server-side components with off-the-shelf services. BaaS enables developers to outsource all the aspects behind a scene of an application so that developers can choose to write and maintain all application logic in the frontend. Examples are remote authentication systems, database management, cloud storage, and hosting.

An example of BaaS can be Google Firebase, a fully managed database that can be directly used from an application. In this case, Firebase (the BaaS services) manages data components on our behalf.

Function-as-a-Service is an environment within which is possible to run the software. Serverless applications are event-driven cloud-based systems where application development relies solely on a combination of third-party services, client-side logic, and cloud-hosted remote procedure calls [1].

FaaS allows developers to deploy code that, upon being triggered, is executed in an isolated environment. Each function typically describes a small part of an entire application. The execution time of functions is typically limited (e.g. 15 min for AWS Lambda). Functions are not constantly active. Instead, the FaaS platforms listen for events that instantiate the functions. Therefore, functions must be triggered by events, such as client requests, events produced by any external systems, data streams, or others. The FaaS provider is then responsible to horizontally scale function executions in response to the number of incoming events.

Serverless applications can be developed in several contexts while, because of its limitations, it might have some issues in other contexts. As an example, long-running functions, such as machine learning training or long-running algorithms might have timeout problems, while constant workloads might result in higher costs compared to indefinitely running on-demand compute services like virtual machines or container run-times. Even if serverless is a very recent topic, researchers already investigated several aspects, such as patterns [12] anti-patterns [10], problems and issues [1].

2.2 Edge Computing

The increasing demand for computation, storage, and network resources are some of the most evident challenges for cloud providers and mobile network operators. Optimizing data traffic has a direct effect on the reliability and quality of services. The cloud has served this purpose for years, however, when it comes to IoT, the cloud falls short. The high number of IoT devices plugged in day induces a high traffic load, which can have a negative effect on the whole network. As a result, Edge Computing has emerged to address these issues, by placing part of the cloud resources (e.g., computation, storage, logic) closer to the edge of the network, which allows faster and more context-dependent data analysis and storage.

In terms of implementation, Edge Computing is composed of a set of nodes, each supports different computation, storage, and network requirements. Different flavors of Edge Computing networks exist, which are similar to what the cloud provides already. Private Edge Computing consists of a private network of Edge Computing nodes managed by a single organization. Public Edge Computing, allows customers to deploy their services on top of a managed infrastructure, and Hybrid Edge Computing, which combines the two previous types.

3 The Serverless Edge Computing Platform

As of today, several edge computing platforms have emerged. Some platforms have been developed for specific purposes, such as increasing the performance of HTTP requests and web content delivery, while others are generic and can be used in different context. In our survey, two main groups of platforms have been identified (Fig. 1). The first group focuses on using serverless functions to customize content at the edge before delivering it to the user, while the second group focuses on executing serverless functions on the data collected at the edge, before either pushing it to the cloud or sending the result back to the user. We have named the first group of platforms *Content Delivery Network platforms*, since they deal mostly with content delivery, while we named the second group *IoT platforms*, since they fit mostly IoT scenarios.

3.1 Content Delivery Network Platforms

Content Delivery Network or Content Distribution Network (CDN) is a network of servers geographically distributed, with the goal of providing high availability and performance by distributing the service closer to the users. CDNs is a very old approach, introduced in the late 1990s to reduce internet bottlenecks [4]. CDN is now frequently adopted by media companies and e-commerce to increase the performances of different services such as video streaming, software downloads, web, and many other systems. Several CDN providers recently saw the potential benefits of providing serverless support in their nodes, enabling not only the caching of the web content on their nodes, but also providing computational capabilities in their nodes, with the serverless technology. In this Section,

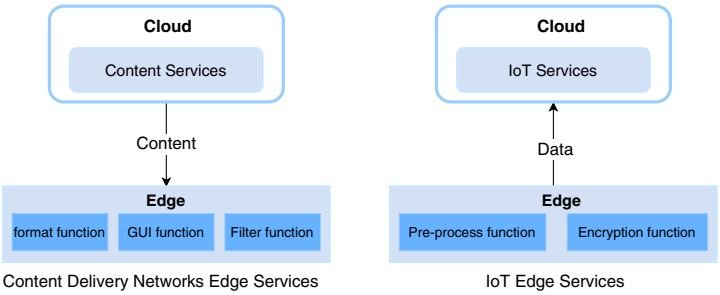

Fig. 1. Serverless at edge platforms categorization

we compare six CDN platforms that allow developing serverless functions on their edge nodes (Table 1).

Akamai Edge

Akamai[1] is one of the leading content delivery network (CDN) providers worldwide. Akamai provides a distributed platform consisting of more than 60,000 servers deployed over 70 countries. Akamai manages more than 15% of the web content. As part of the provided services, a dedicated edge platform called Edgeworkers has been developed. The main goal of edge workers is to allow cloud platforms to provide personalized business logic at the edge to support context aware services and at the same time reduce services latency. In this context, serverless functions can be customized and deployed closer to the customer infrastructure. Developers can take advantage of the wide network managed by Akamai to have control over where the functions are needed and what type of customization is needed to improve user experience in terms of performance and content.

IBM Edge Functions

Edge Functions on IBM Cloud Internet Services (CIS)[2] supports serverless computing at the edge closer to end-users across 180+ global network points of presence. As an example, it is designed to be able to pre-process HTTP requests and post-process responses e.g. for personalized user experience or improved API responsiveness. It is based on "isolates" that run on the V8 engine thus limiting the development to JavaScript.

Cloudflare

Cloudflare[3] is a CDN provider, with the main focus on performances. Cloudflare handles nearly 10% of the Internet HTTP requests, with peaks of more than 25

[1] https://www.akamai.com/us/en/products/performance/serverless-computing-edgeworkers.jsp.
[2] https://cloud.ibm.com/docs/infrastructure/cis?topic=cis-edge-functions.
[3] https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing.

trillion monthly requests through their network. Cloudflare provides servers in 154 locations around the world.

Similarly to Akamai, it provides workers (Cloudflare workers) that enable developers to run JavaScript code as serverless functions on the edge nodes, while it controls the location of the edge nodes depending on the request locations.

Cloudfront

Cloudfront[4] is a CDN that acts as a distributed cache for web applications, part of the Amazon Web Services (AWS) offer. Cloudfront fetches files from their source location ("origin" in CloudFront terms) and places the copies of the files in different edge locations across the Americas, Europe, Asia, Africa, and Oceania. It enables to deploy serverless functions in its edge nodes, using the AWS "Lambda@Edge" functions enabling to run business logic, implemented in the functions. Differently from Akamai and Cloudflare, it supports the deployment of functions in several languages.

Edjx

Edjx[5] is a distributed edge computing platform. Combining packed as small microservers, container technology, and blockchain, Edjx can deliver a rich environment for developers to write, test and deploy serverless functions at the edge. The main target of Edjx is IoT applications with high bandwidth and low latency requirements. Using blockchain, Edjx provides a Pay as you go model for resources provisioning. Two main components make up the Edjx infrastructure:

- EDJX Nanoserver Infrastructure: it represents the hardware back-end of the system. It is composed of a set of lightweight servers that can be deployed as edge nodes. The servers are packed as small form factor with an Intel i7 CPU, 16 GB of RAM and 1 TB of storage.
- EDJX Serverless Edge: it represents the software component to manage the serverless functions lifecycle and orchestration. Since Edge Computing nodes need to collaborate in order to deliver services, the platform creates a peer-to-peer network among all the participating nodes.

Edjx promise is to make the deployment process transparent to the developers. The platform handles the process of locating the closest node to the user and deploy the serverless functions. To securely access services and data records, Edjx relies on Chainyard[6] to deliver blockchain based distributed serverless applications.

[4] https://aws.amazon.com/cloudfront/.
[5] https://edjx.io.
[6] https://chainyard.com/.

Stackpath

Stackpath[7] is a general-purpose cloud-based CDN with edge nodes in the whole world except Canada, Russia and Africa. The Stackpath serverless scripting engine is built on the Chrome V8 JavaScript Engine providing support for JavaScript. However, it enables also us to use WebAssembly supporting additional language support such as PHP, C, C++, Go, Python, Perl, Rust, and more.

Table 1. Comparison of the CDN platforms

Features	CDN platforms with serverless support						
	Akamai	Cloudflare	Stackpath	CloudFront	Edjx	IBM edge functions	Nuclio
Support of AI on the edge				✓	✓		✓
Availability	Globally	Globally	Limited	Globally	Limited	Globally	Limited
Supported platforms (edge hardware)	Akamai nodes	Cloudflare nodes	Stackpath nodes	AWS nodes	Nanoservers	IBM centers	Portable across constrained devices
Supported languages	JavaScript	JavaScript	Multi-lang	Multi-lang	Multi-lang	JavaScript	Multi-lang
Cost model	Pay as you go						Hosting cost
License	Proprietary						Open source

3.2 IoT Platforms

Internet of Things (IoT) platforms include connectivity, management, and programming mean for running various devices or things as part of Internet applications. In their simplest form, IoT devices transmit some sensor readings but as more advanced they include various functionalities such as preprocessing of sensor data or actuating with the physical world. These advanced IoT devices can thus be seen as an extension of the Internet-based application system including application software connecting with both cloud and edge components.

AWS IoT Greengrass

AWS IoT Greengrass[8] is Amazon's extension of the cloud to the edge of the network and physical devices. Greengrass has been designed from the beginning for use in the user's own hardware while using the same cloud management mechanisms, analytics, and durable storage. Regarding serverless Greengrass is well known for its capability to execute AWS Lambda functions and in most cases, they can be the same as those run in the cloud. The Lambdas that can be run on Greengrass edge devices can be implemented in several programming languages and the edge software platform can be installed on platforms including x86-64, ARMv8, ARMv7 and also as Docker containers.

[7] https://www.stackpath.com/products/edge-computing/serverless-scripting/.
[8] https://aws.amazon.com/greengrass/.

Azure IoT Edge
Azure IoT Edge[9] is Microsoft's edge computing and IoT Hub cloud extension for the physical devices of the user. IoT Edge supports several Linux versions and Windows 10 or Windows Server 2019 on their Tier 1 level and multiple other operating systems including virtual machines as Tier 2 level supported. X86-64 as well as 32-bit and 64-bit ARM architectures are supported. In terms of serverless functionality, it allows the containerization of Azure Functions developed in multiple programming languages to be deployed on IoT Edge devices. It is worth noting that IoT Edge software is free and open source.

Fogflow
Fogflow[10] is an edge computing framework designed to automate and optimize IoT services orchestration. It leverages three types of context to provide unique context-driven feature, *i)* System context: it relies on geo-distributed services to make sure that resources are available where needed, *ii)* Data context: it uses a unified data model to detect relations between tasks in order to optimize task flows, and *iii)* Usage context: orchestration decisions can be based on user-specific rules and thresholds. In Fogflow, the flow of execution can span across multiple Edge Computing nodes depending on the different combinations of aforementioned types of context (e.g., two services located in different areas and the second service relies on the first service output). To facilitate services migration, Docker containers are used to package services logic and all its dependencies. On top of Fogflow, serverless functions can be deployed. Fogflow support serverless function by:

- invoking serverless function once the input data are available
- automatically managing scalability of instances (e.g., create new instances)
- automatically locating the best Edge Computing node (i.e., closer to the data producer or data consumer) to deploy serverless functions.

Nuclio
Nuclio[11] is a serverless framework focusing on high data, I/O and compute intensive workloads. The framework supports a wide range of data sources and supports CPU and GPU execution modes. One of the main goals of Nuclio is to provide an open environment that allows easy portability and rapid deployment time. It supports most popular data science tools such as Jupyter and kubeflow, which increases deployment automation. Nuclio has been used predominantly in IoT scenarios where IoT data can be analysed closer to the data sources.

[9] https://docs.microsoft.com/en-us/azure/iot-edge/.
[10] https://github.com/smartfog/fogflow.
[11] https://nuclio.io.

OpenWhisk-Light

The standard OpenWhisk[12] is an open-source initiative for distributed serverless execution of functions in response to various events. OpenWhisk-Light[13] is a runtime with the standard OpenWhisk API for local or edge execution also supporting resource-constrained devices while maintaining a centralized OpenWhisk cloud instance as a master repository and catalog of its actions (i.e. functions). It supports the execution of OpenWhisk actions developed using multiple programming languages and can be deployed on the edge as Docker containers. It has also been demonstrated working on devices as constrained as a Raspberry Pi which makes it a candidate for IoT edge devices. It is based on an open source licensing similar to the original OpenWhisk (Table 2).

Table 2. Comparison of the IoT platforms

Features	IoT platforms with serverless support				
	AWS GreenGrass	Azure IoT	FogFlow	OpenWhisk-Light	Nuclio
Support of AI on the edge	✓	✓	✓	✓	✓
Availability	Globally	Globally	Limited	Limited	Limited
Edge hardware	Docker support	Tier 1: containers support. Tier 2: Virtual machines support	Docker support	containers support. Demonstrated for limited operation also in Raspberry Pi	Portable across constrained devices
Supported languages	Multi-lang	Multi-lang	Multi-lang	Multi-lang	Multi-lang
Cost model	Pay as you go		Private setting		Private or hosting cost
License	Proprietary	Open source			

4 Discussion

The initial comparison suggests that the existing platforms in the two categories have clearly specific goals. While the CDN category focuses more on taking advantage of serverless technology to increase availability and reduce costs, the IoT category points more towards portability, AI and multi-language support (Fig. 2).

Even-though serverless on the edge is still at its infancy, the first proofs of its potential usage can already be seen in the proposed solutions and platforms. On one side, existing Edge providers are extending their offers providing serverless support on their edge nodes. On the other side, new serverless-specific edge platforms have been introduced in the last years.

[12] https://openwhisk.apache.org.
[13] https://github.com/kpavel/openwhisk-light.

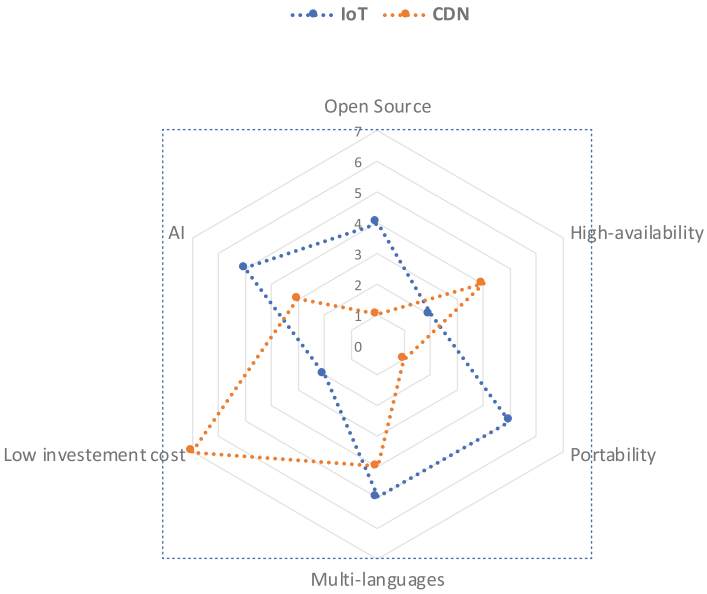

Fig. 2. Serverless at edge categories comparison

Existing edge platforms often enable only to deploy functions written with a limited set of languages. As an example, the traditional CDN platforms enable to write Javascript code on their edge nodes, while new platforms enable developers to use different languages.

IoT applications with more advanced processing on the edge device or edge of the network could significantly benefit from the serverless paradigm and especially the management and deployment of versions across fleets of devices. In addition to traditional sensor data processing, video or image-based processing as well as distributed AI-based inference are expected to be application areas of interest.

For IoT targeted solutions it seems that Microsoft with its IoT Edge is striving for a more open platform compared to AWS Greengrass. Both platforms support different hardware and installation on own equipment but the biggest difference is in Microsoft IoT Edge open source licensing that enables companies to use and extend their open source components on local hardware. Microsoft also supports an open ecosystem through the Azure Marketplace, e.g. acquiring solutions developed by others and deploying on the edge. Both of the platforms, however, rely heavily on their cloud service counterparts increasing the vendor lock-in. The OpenWhisk-Light is a fully open source alternative that offers similar features but with less tooling and support. As a consequence, however, it requires management of the OpenWhisk cloud counterpart to which the edge component is an extension of.

4.1 Open Issues

This work enabled us to identify a set of open issues:

- *Vendor Lock-In.* Commercial serverless platforms require to write functions that use the infrastructure provided, increasing vendor lock-in. As an example, an application developed with Greengrass would require a major effort to be deployed in Azure IoT. Currently, no frameworks allow to use hybrid clouds and to write generic functions that could be deployed in different ecosystems.
- Lack of decision frameworks to understand when is beneficial or not to use serverless on edge
- Lack of best practices, patterns and anti-patterns for creating serverless applications on the edge.

We believe that the research community should help practitioners to understand how to create serverless functions on the edge that could be deployed everywhere, and provide guidelines, including validated patterns and anti-patterns for creating serverless applications on the edge.

5 Conclusion

In this paper, we described the most common platforms for Serverless in Mobile Edge Computing.

Some of the selected platforms are targeted to specific purposes such as IoT, while others are specifically targeting Content Delivery Network (CDN). Moreover, it is interesting to note that several CDN providers that offered edge support for increasing the performances of web systems recently introduced the possibility to deploy code as serverless functions, enabling to compose dynamic web pages on the edge, but also to run part of the business logic.

As future work, we are planning to investigate the usefulness of serverless on edge computing, with a special focus on the identification of benefits and issues in this context and supporting companies to understand when it is beneficial to adopt it, and when it would be better to use different solutions.

References

1. Baldini, I., et al.: Serverless computing: current trends and open problems. In: Chaudhary, S., Somani, G., Buyya, R. (eds.) Research Advances in Cloud Computing, pp. 1–20. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5026-8_1
2. Baresi, L., Mendonça, D.F.: Towards a serverless platform for edge computing. In: Proceedings of the IEEE International Conference on Fog Computing (ICFC 2019), pp. 1–10. IEEE (2019)
3. Cheng, B., Fuerst, J., Solmaz, G., Sanada, T.: Fog function: serverless fog computing for data intensive IoT services. In: Proceedings of the IEEE International Conference on Services Computing (SCC 2019). pp. 28–35. IEEE (2019)

4. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Weihl, B.: Globally distributed content delivery. IEEE Internet Comput. **6**(5), 50–58 (2002). https://doi.org/10.1109/MIC.2002.1036038
5. Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I., Imran, M.: The role of edge computing in internet of things. IEEE Commun. Mag. **56**(11), 110–115 (2018). https://doi.org/10.1109/MCOM.2018.1700906
6. Hästbacka, D., et al.: Dynamic and flexible data acquisition and data analytics system software architecture. In: 2019 IEEE SENSORS, pp. 1–4 (2019). https://doi.org/10.1109/SENSORS43011.2019.8956662
7. Liu, M., Yu, F.R., Teng, Y., Leung, V.C.M., Song, M.: Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing. IEEE Trans. Wirel. Commun. **18**(1), 695–708 (2019). https://doi.org/10.1109/TWC.2018.2885266
8. Nastic, S., et al.: A serverless real-time data analytics platform for edge computing. IEEE Internet Comput. **21**(4), 64–71 (2017). https://doi.org/10.1109/MIC.2017.2911430
9. Ning, H., Li, Y., Shi, F., Yang, L.T.: Heterogeneous edge computing open platforms and tools for internet of things. Future Gener. Comput. Syst. **106**, 67–76 (2020). https://doi.org/10.1016/j.future.2019.12.036
10. Nupponen, J., Taibi, D.: Serverless: what it is, what to do and what not to do. In: IEEE International Conference on Software Architecture (ICSA 2020) (2020)
11. Palade, A., Kazmi, A., Clarke, S.: An evaluation of open source serverless computing frameworks support at the edge. In: Proceedings of the IEEE World Congress on Services (SERVICES 2019), vol. 2642–939X, pp. 206–211 (2019). https://doi.org/10.1109/SERVICES.2019.00057
12. Taibi, D., El Ioini, N., Pahl, C., Schmid Niederklfler, J.R.: Serverless cloud computing (function-as-a-service) patterns: a multivocal literature review. In: International Conference on Cloud Computing and Services Science (CLOSER 2020) (2020)
13. White, G., Cabrera, C., Palade, A., Clarke, S.: Augmented reality in IoT. In: Liu, X., et al. (eds.) ICSOC 2018. LNCS, vol. 11434, pp. 149–160. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17642-6_13